Self-Error-Instruct: Generalizing from Errors for LLMs Mathematical Reasoning

Erxin Yu[1], Jing Li[1,2]*, Ming Liao[1], Qi Zhu[3], Boyang Xue[4], Minghui Xu[3], Baojun Wang[3], Lanqing Hong[3], Fei Mi[3], Lifeng Shang[3]
[1]Department of Computing, The Hong Kong Polytechnic University
[2]Research Centre for Data Science & Artificial Intelligence
[3]Huawei Noah's Ark Lab, [4]The Chinese University of Hong Kong
erxin.yu@connect.polyu.hk, jing-amelia.li@polyu.edu.hk

Abstract

Although large language models demonstrate strong performance across various domains, they still struggle with numerous bad cases in mathematical reasoning. Previous approaches to learning from errors synthesize training data by solely extrapolating from isolated bad cases, thereby failing to generalize the extensive patterns inherent within these cases. This paper presents Self-Error-Instruct (SEI), a framework that addresses these model weaknesses and synthesizes more generalized targeted training data. Specifically, we explore a target model on two mathematical datasets, GSM8K and MATH, to pinpoint bad cases. Then, we generate error keyphrases for these cases based on the instructor model's (GPT-4o) analysis and identify error types by clustering these keyphrases. Next, we sample a few bad cases during each generation for each identified error type and input them into the instructor model, which synthesizes additional training data using a self-instruct approach. This new data is refined through a one-shot learning process to ensure that only the most effective examples are kept. Finally, we use these curated data to fine-tune the target model, iteratively repeating the process to enhance performance. We apply our framework to various models and observe improvements in their reasoning abilities across both in-domain and out-of-domain mathematics datasets. These results demonstrate the effectiveness of self-error instruction in improving LLMs' mathematical reasoning through error generalization.

1 Introduction

Large language models (LLMs) (Brown et al., 2020; Ouyang et al., 2022; Jiang et al., 2023; Team, 2024) have demonstrated remarkable capabilities across various domains, particularly after instruction-based fine-tuning. Yet, LLMs are still facing substantial challenges in complex reasoning tasks, particularly in mathematical reasoning. They continue to encounter numerous bad cases, often committing errors that compromise their reliability.

Previous work has taken advantage of these errors to improve model performance. Mistake-tuning and self-rethinking (Tong et al., 2024b) leverage the historical errors of LLMs to enhance their performance during both the fine-tuning and inference stages. LLMs like ChatGPT (Ouyang et al., 2022) are utilized to synthesize training datasets based on the bad cases from smaller models (Ying et al., 2024; Tong et al., 2024a). LLMs are also employed to optimize the reasoning steps of smaller models (An et al., 2024), generating corrective data to train these models.

However, current methods predominantly synthesize training data from individual bad cases. While this can somewhat enhance model performance, the data often suffers from a lack of generalization because it is too reliant on specific instances, which limits its ability to cover a wider array of error patterns. To overcome this limitation, we introduce the Self-Error-Instruct (SEI) framework, which aims to generalize training data based on error types instead of focusing solely on individual cases. For example, in Figure 1, the left subfigure displays various error types of Qwen2.5-Math. We enhanced its mathematical reasoning by generalizing the data according to these error types, which is depicted in the right subfigure. To the best of our knowledge, *we are the first to explore data synthesis and selection for LLMs to generalize from errors based on error types in math reasoning.*

Specifically, we begin by assessing target model to identify bad cases. An instructor model is first used to pinpoint errors from these bad cases and generate relevant keyphrases, then cluster these keyphrases into distinct error types. We select a few samples from each error type as prompts for the instructor model in a self-instruct manner to synthesize new data. We further apply a one-shot

*Corresponding author

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 8504–8519
July 27 - August 1, 2025

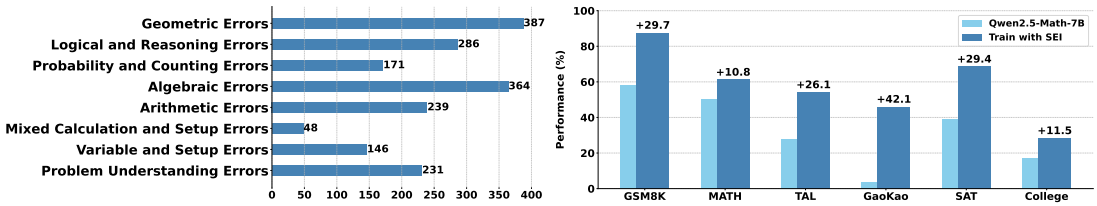

Figure 1: The left table shows some error types of Qwen2.5-Math-7B on Math and GSM8K training set, while the right presents the results after training on data generalized from error categories.

learning-based refinement to the new data to verify its effectiveness to rectify the target model's deficiencies while maintaining the target model's current success, only keeping the data that works. This refinement process is iteratively repeated to improve the model's performance.

We employ LLaMA3-8B-Instruct, Qwen2.5-Math-7B, and Mathstral-7B-v0.1 as the target models to identify bad cases within the training datasets, GSM8K and MATH. We conduct comprehensive evaluations using both in-domain and out-of-domain testing. For in-domain tests, we use test sets from GSM8K and MATH. For out-of-domain tests, we utilize four additional mathematical reasoning datasets: TAL, GaoKao, SAT, and College.

Experimental results show that training the target models with our synthesized data significantly improves performance on both in-domain and out-of-domain test sets. Specifically, LLaMA3 and Mathstral achieve average improvements of 1.72% and 0.98%, respectively, while Qwen2.5 shows a more significant gain of 24.94%. Additionally, our one-shot learning-based data selection method is highly effective, outperforming both random selection and LESS (Xia et al., 2024), a recently proposed gradient-based data selection method. It also surpasses the performance of models trained on the full dataset. This demonstrates that our approach can accurately identify high-quality training data to enhance model performance. Our experiments further highlight the importance of resolving bad cases in the one-shot learning selection process and maintaining the model's correctness on the original good cases. Finally, we analyze the fix rate of bad cases at each iteration, examine the impact of generalized data volume on model performance, and compare two training strategies: iterative training with data synthesized in each round versus training from scratch with all synthesized data. In summary, our contributions are as follows:

- We improve data generalization by organizing mathematical reasoning data according to error types instead of individual bad cases.
- We propose the Self-Error-Instruct framework, which analyzes bad cases through keyphrases extraction and clustering, then performs data generalization for each cluster.
- Experiments show that our method efficiently generalizes data based on error types, enhancing mathematical reasoning skills and validating the effectiveness of our data selection strategy.

2 Related Work

2.1 Mathematical Reasoning

With the rapid advancement of large language models, they have shown remarkable capabilities across a wide range of NLP tasks, as demonstrated by models like ChatGPT (Ouyang et al., 2022), Claude (Anthropic, 2024), and Gemini (Team, 2024). However, mathematical reasoning remains a significant challenge for these models. To address this issue, many models, such as OpenAI o1 (OpenAI, 2024), Qwen-2.5-Math (Yang et al., 2024), and DeepSeek-Math (Shao et al., 2024), have undergone specialized training for mathematical tasks. Researchers have explored various strategies to enhance performance in this area, including prompting, pretraining, and fine-tuning.

Among these techniques, some focus specifically on learning from errors to enhance model performance. LEMA (An et al., 2024) leveraged GPT-4 (OpenAI, 2024a) to correct the model's erroneous reasoning paths and used the refined reasoning paths to fine-tune the model. Self-rethinking and mistake tuning (Tong et al., 2024b) analyze the causes of model errors to improve reasoning performance. The former uses an iterative process to help the model avoid repeating past mistakes, while the latter fine-tunes the model by incorporating correct and erroneous reasoning examples. LLM2LLM (Tong et al., 2024a) generates new synthetic data based on error cases to improve model performance iteratively. Learning from error and learning from

error by contrast (Ying et al., 2024) are two strategies designed to improve the performance of target models. The former generates targeted training data by analyzing erroneous responses, while the latter by contrasting correct and incorrect responses. In contrast to these approaches, which focus solely on individual bad cases, our method generalizes data based on error types. This allows for more systematic coverage of diverse issues, enhances data diversity, and improves generalization ability.

2.2 Data Selection

Data selection plays a crucial role in instruction tuning, as it helps identify high-quality data, enhancing model performance and generalization while minimizing noise to optimize training. LIMA (Zhou et al., 2023) achieved exceptional performance by selecting 1,000 high-quality question-answer pairs for instruction tuning, delivering results comparable to those obtained through large-scale instruction tuning and reinforcement learning. Instruction-following difficulty (Li et al., 2024a) was proposed to evaluate the difficulty of following instructions for each sample. LESS (Xia et al., 2024) identified training data most similar to the validation set based on gradient features. NUGGETS (Li et al., 2024b) assessed the impact of candidate instructions on a predefined task set's perplexity using one-shot learning, comparing the score differences between zero-shot and one-shot learning as a reference for data selection. Building on NUGGETS, we designed a one-shot learning data selection method tailored for mathematical reasoning. This method selects data based on whether the generated data can address the target model's bad cases while preserving its good cases.

3 Our Self-Error-Instruct Framework

Our framework [1] aims to enhance the mathematical reasoning ability of the target model $\mathbf{M}_{\text{target}}$ by identifying its weaknesses, referred to as bad cases, on an existing mathematical training dataset $\mathbf{D}_{\text{train}}$. These bad cases are analyzed to guide the synthesis of targeted training data that directly addresses the model's specific shortcomings. By progressively training on this tailored data, the mathematical capabilities of $\mathbf{M}_{\text{target}}$ are effectively improved.

As shown in Figure 2, our process consists of four key steps: 1) **Bad Case Extraction** (Section 3.1), which identifies the incorrect cases where the target model $\mathbf{M}_{\text{target}}$ fails on the existing mathematical reasoning dataset $\mathbf{D}_{\text{train}}$. 2) **Self Error Instruct** (Section 3.2) generates targeted data for $\mathbf{M}_{\text{target}}$ by first identifying error keyphrase, then clustering similar errors, and finally synthesizing data specifically tailored to address the identified error types. 3) **Data Selection** (Section 3.3) filters and selects high-quality data from the generated dataset, ensuring that only the most relevant and effective examples are used for training. 4) **Iterative Training** (Section 3.4) uses the selected data to retrain $\mathbf{M}_{\text{target}}$, iterating this process to continuously refine and enhance the model's performance, thereby improving its mathematical reasoning capabilities with each cycle.

3.1 Bad Case Extraction

For each problem with its correct reasoning path (q_i, r_i) in the training dataset $\mathbf{D}_{\text{train}}$, we use $\mathbf{M}_{\text{target}}$ to generate a reasoning path. During this process, we identify and collect the bad case $(q_i, r_i, \hat{r_i})$ into the error dataset $\mathbf{D}_{\text{error}}$, where the answers derived from the reasoning paths differ, i.e., $\mathbf{Ans}(\hat{r_i}) \neq \mathbf{Ans}(r_i)$, where $\mathbf{Ans}(\cdot)$ is the function that extracts the answer from a given reasoning path. Thus, the error dataset is defined as:

$$\mathbf{D}_{\text{error}} = \{(q_i, r_i, \hat{r_i}) \mid \mathbf{Ans}(\hat{r_i}) \neq \mathbf{Ans}(r_i)\}. \quad (1)$$

3.2 Self Error Instruct

In this phase, for each bad case in $\mathbf{D}_{\text{error}}$, we leverage the $\mathbf{M}_{\text{instructor}}$ model to perform error analysis by examining the reasoning paths and generating an error keyphrase that captures the nature of the mistake. These error keyphrases are then clustered into distinct groups based on similarity. For each error type, targeted data synthesis generates new training samples specifically designed to address model weaknesses. This process produces the curated dataset $\mathbf{D}_{\text{SEI}}$, containing diversity and error-specific training samples to enhance the target model's reasoning ability.

Error Keyphrase Generation. During this stage, we address each bad case $(q_i, r_i, \hat{r_i})$ in the dataset $\mathbf{D}_{\text{error}}$ using the $\mathbf{M}_{\text{instructor}}$ model for detailed error analysis. This process generates an error keyphrase e_i, which captures the specific nature of the error. To achieve this, we employ a structured function $\mathbf{Extract}[\cdot]$ with a keyphrase extraction prompt to analyze the incorrect reasoning path $\hat{r_i}$ and produce

[1] Our code is available at https://github.com/ErxinYu/SEI.

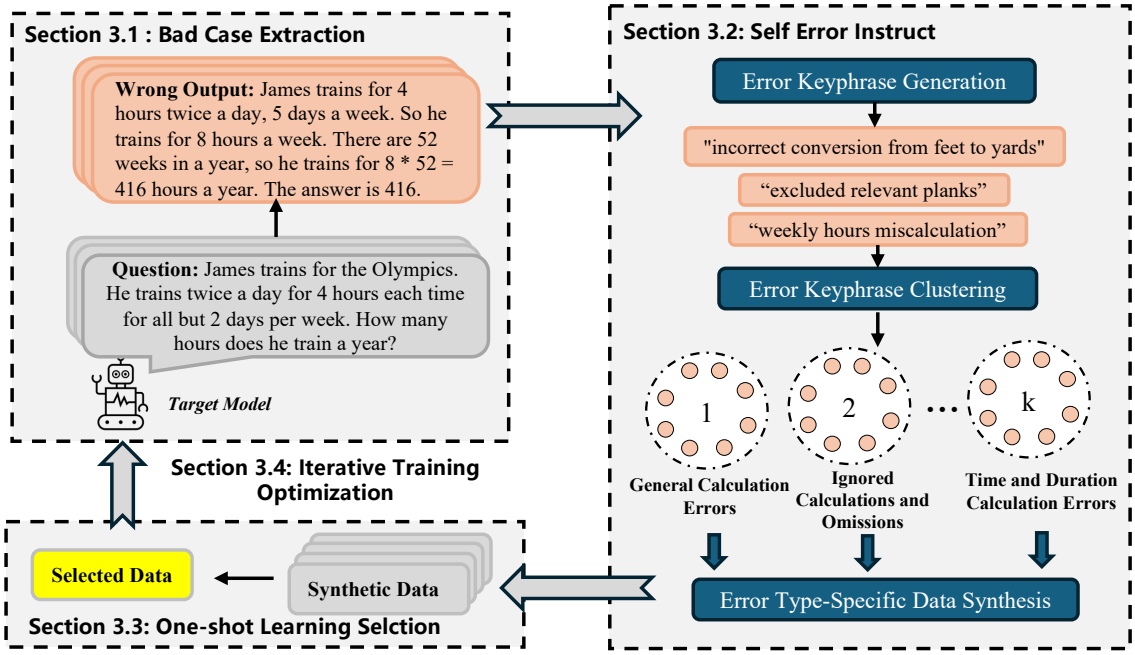

Figure 2: An overview of our Self-Error-Instruct framework. It consists of four key steps: (1) **Bad case extraction** identifies failure cases from the target model. (2) **Self-error-instruct** generates error keyphrases, clustering, and synthesizes data for each error type. (3) **One-shot learning data selection** retains only high-quality and effective examples for training. (4) **Iterative training** refines the target model by fine-tuning it with the curated data and repeating the process to further improve performance.

the corresponding error keyphrase. Details of the prompt are provided in the Appendix A.2. The process is mathematically represented as follows:

$$\textit{EK-Set} = \{e_i \mid e_i = \mathbf{Extract}[\mathbf{M}_{\text{instructor}}, (q_i, r_i, \hat{r}_i)], \forall (q_i, r_i, \hat{r}_i) \in \mathbf{D}_{\text{error}}\}, \quad (2)$$

where *EK-Set* represents the collection of error keyphrases generated for all bad cases in $\mathbf{D}_{\text{error}}$. This approach ensures that each e_i accurately captures the underlying issue in the model's reasoning path, providing a solid foundation for subsequent clustering and data synthesis steps.

Error Keyphrases Clustering. After obtaining the *EK-Set*, we utilize the $\mathbf{M}_{\text{instructor}}$ model to cluster the keyphrases within this set. This clustering process identifies distinct error types, denoted as the *ET-Set*. The process can be mathematically expressed as:

$$\textit{ET-Set} = \mathbf{Cluster}[\mathbf{M}_{\text{instructor}}, \textit{EK-Set}], \quad (3)$$

where $\mathbf{Cluster}[\cdot]$ is a clustering prompt (see Appendix A.3) designed to group the error keyphrases into coherent and distinct types. Each type is manually reviewed (see Appendix C) to filter and validate its relevance and appropriateness.

Error Type-Specific Data Synthesis. For each error type within the *ET-Set*, we begin by sampling a subset of bad cases from the same error type, which serve as in-context learning prompts. These prompts are then used to guide $\mathbf{M}_{\text{instructor}}$ in generating additional data that falls under the same error type. This process ensures that the generated data remains consistent with the specific error patterns of the given type, thereby expanding our dataset with more diverse but relevant examples. Through this process, we ultimately obtain a synthesized dataset $\mathbf{D}_{\text{SEI}}$, which enriches our data with examples covering distinct error patterns. The specific prompt used for this generalization process can be found in the Appendix A.4.

3.3 One-shot Learning Selection

After obtaining the generalized dataset $\mathbf{D}_{\text{SEI}}$ targeting specific errors, our goal is to select a small subset of high-quality data for training the target model. In previous work, NUGGETS (Li et al., 2024b) uses a one-shot learning approach to filter data. It calculates a score for each instruction example based on its impact on the perplexity of a set of pre-defined tasks, allowing for the identification of the most beneficial data for instruction tuning.

In our approach to mathematical reasoning tasks, instead of relying on perplexity, we directly evaluate whether the newly generalized data can effectively serve as a one-shot prompt to guide the target

model in resolving bad cases. Furthermore, we aim to ensure that the target model maintains its performance on good cases originally answered correctly, preserving its effectiveness across challenging and straightforward examples. First, we randomly sample a subset of bad cases and good cases to create a validation set, $\mathbf{D}_{\text{dev}}$. Next, we evaluate each sample in $\mathbf{D}_{\text{SEI}}$ by measuring the number of cases in $\mathbf{D}_{\text{dev}}$ that can be resolved when the sample is used as a one-shot prompt. This evaluation serves as the criterion for selecting high-quality data. The process can be represented as:

$$r_i^j = M_{\text{target}}(\underbrace{q^j r^j}_{\text{One-Shot Prompt}} \oplus q_i) \quad (4)$$

$$S_{\text{osl}}^j = \sum_i \mathbb{I}[Ans(r_i^j) = Ans(r_i)] \quad (5)$$

The expression $q^j r^j$ represents the j-th synthetic data point from the dataset $\mathbf{D}_{\text{SEI}}$. The score S_{osl}^j is the one-shot learning score, calculated by summing the indicator function $\mathbb{I}[\cdot]$, which is 1 if the answer from r_i^j matches r_i, and 0 otherwise. Here, $q_i r_i$ are elements from $\mathbf{D}_{\text{dev}}$, where r_i is the correct reasoning path for q_i. The prompt for one-shot learning is shown in Appendix 7. For each synthetic data in $\mathbf{D}_{\text{SEI}}$, calculate the set of one-shot learning scores $\{S_{\text{osl}}^1, S_{\text{osl}}^2, \ldots, S_{\text{osl}}^m\}$. By sorting these scores, we obtain the selection $\mathbf{D}_{\text{SEI}}^{\text{osl}}$.

3.4 Iterative Training Optimization

The selected data, $\mathbf{D}_{\text{SEI}}^{\text{osl}}$, is used to train the target model, $\mathbf{M}_{\text{target}}$. After the model is enhanced through this training, it is applied to $\mathbf{D}_{\text{train}}$ once more to identify new bad cases that it still struggles with. This process is iterated, continuously optimizing the target model by improving its ability to handle challenging examples, thereby enhancing its overall mathematical reasoning ability.

4 Experimental Setup

4.1 Data Synthetic

We identify bad cases from the training datasets of GSM8K and MATH, using GPT-4o [2] (OpenAI, 2024b) as the instructor model to generate error keyphrases, perform clustering, and synthesize data. For each error type, during the self-error instruct process, we sample 5 data points from the error dataset $\mathbf{D}_{\text{error}}$ and 3 data points from the already generated data within the current error type to serve as prompts. Each time, GPT-4o generalizes 20 new math data. We then filter out data with a Rouge-L score greater than 0.7 compared to the GSM8K and MATH training and test datasets to enhance diversity and prevent test set leakage. We randomly select 100 data points, comprising 50 good and 50 bad cases, to construct the validation set $\mathbf{D}_{\text{dev}}$. The number of iterations for data synthesis and model training is 3. In each iteration, we generate 10,000 data points by synthesizing 5,000 examples for the error types of GSM8K and 5,000 for MATH. We select the top 5% of the synthetic data from each part and combine them into a unified dataset for training. Over three iterations, we generate a total of 30,000 data points and select 1,500 for training. We also compared two methods for training the target model: iterative training, which starts from the model trained in the previous round, and training from scratch, which uses the selected data in a single step. The results of these two methods are shown in Table 5.

[2]We use the Microsoft Azure AI services at https://azure.microsoft.com/

Dataset	Difficulty	Difficulty	Train	Test
GSM8K	Elementary	Easy	7,473	1,319
MATH	Competition	ExHard	7,498	5,000
TAL-SCQ	K12 Math	Medium	-	1,496
GaoKaoBech-Math	High School	Hard	-	508
SAT-MATH	High School	Hard	-	102
CollegeMath	College	ExHard	-	2,818

Table 1: Statistics of Different Datasets. We extract bad cases from the GSM8K and MATH training sets and use the test sets of all datasets for evaluation. Datasets marked with "-" indicate only test data is available and are used for out-of-domain evaluation.

4.2 Target Model Setting

We use the instruction-tuned Llama3-8b-instruct model (Grattafiori et al., 2024), the math-specialized Qwen2.5-Math-7B (Yang et al., 2024), and Mathstral-7B-v0.1 (Jiang et al., 2023) as our target models. During training, we employ LoRA (Hu et al., 2021) with a maximum sequence length of 2048 tokens, set the number of training epochs to 3, and use a learning rate 2e-05.

4.3 Evaluation

We used the GSM8K (Cobbe et al., 2021) and Math (Hendrycks et al., 2021) test sets for in-domain evaluation. For out-of-domain evaluation, we utilized four challenging datasets: 1) **TAL-SCQ** (TAL, 2023): A K-12 mathematics test set containing 1,496 test examples. 2) **GaoKaoBench-Math** (Zhang et al., 2024): Comprising 508 test exam-

Models	In-Domain		Out-of-Domain				AVG
	GSM8K	MATH	TAL	GaoKao	SAT	College	
Llama3-8B-Instruct	77.56	27.36	37.03	15.55	39.22	15.54	35.38
+ Training data	63.99	23.32	29.01	12.00	34.31	13.41	29.34
+ Bad Cases	65.13	23.20	30.08	11.22	33.33	13.41	29.40
+ Self-Instruct	74.83	26.20	35.44	14.76	37.25	15.26	33.96
+ LLMs-as-Instructors	79.37	27.84	36.17	16.14	38.24	**15.79**	35.59
+ LLM2LLM	76.61	27.60	**40.10**	15.16	38.24	15.51	35.54
+ SEI-ICL	**79.76**	**28.42**	39.91	**16.73**	**42.15**	15.61	**37.10**
Qwen2.5-Math-7B	57.92	50.52	28.07	3.93	39.22	16.96	32.77
+ Training data	57.54	56.22	46.19	38.78	65.69	24.20	48.10
+ Bad Cases	64.21	56.90	45.45	34.44	63.73	22.36	47.85
+ Self-Instruct	80.57	58.24	52.66	43.31	65.69	26.87	54.56
+ LLMs-as-Instructors	79.31	58.76	**54.62**	45.43	63.73	28.07	54.99
+ LLM2LLM	81.17	58.88	53.56	43.11	65.69	27.96	55.06
+ SEI-ICL	**87.64**	**61.28**	54.21	**46.06**	**68.62**	**28.42**	**57.71**
Mathstral-7B-v0.1	80.67	52.58	48.66	47.83	61.76	**25.80**	52.88
+ Training data	72.10	44.40	41.44	42.91	56.86	24.17	46.98
+ Bad Cases	70.58	46.06	41.24	43.11	59.80	24.59	47.56
+ Self-Instruct	79.68	52.02	47.13	44.69	58.82	25.28	51.27
+ LLMs-as-Instructors	79.61	52.42	48.13	43.31	**63.73**	25.19	52.07
+ LLM2LLM	81.35	52.64	46.79	45.87	59.08	25.16	51.82
+ SEI-ICL	**82.87**	**53.70**	**49.47**	**48.62**	62.75	25.72	**53.86**

Table 2: Main results on in-domain and out-of-domain mathematical test sets, evaluated using the exact match (EM). All experiments are conducted in a zero-shot setting. SEI-ICL refers to our proposed method, which leverages the self-error-instruct framework to generalize and train using the top 5% of data selected through one-shot learning. For fair comparison, the generalized data sizes for the baselines are kept consistent with SEI-ICL.

ples, this dataset features math problems from the Chinese high-school curriculum. 3) **SAT-MATH** (Zhong et al., 2024): Consisting of 102 questions, this dataset includes math problems from the U.S. high-school curriculum. 4) **CollegeMath** (Tang et al., 2024): This dataset contains 2,818 test examples of college-level math problems. The detailed dataset statistics are provided in Table 1.

We evaluated the models on these datasets using greedy decoding in a zero-shot setting, with the maximum generation length set to 2048. Performance was measured using Exact Match (EM), where answers were extracted from the generated reasoning paths and compared to the correct ones. All evaluations were conducted using the MWP-Bench framework [3].

4.4 Baselines

We compare with several baselines: 1) **Training Data**, where the model is trained on the combined GSM8K and MATH datasets; 2) **Bad Cases**, using bad cases from the initial target model; 3) **LLMs-as-Instructors**, using Learning from error (LE) by generating tailored training data for errors. (Ying et al., 2024) 4) **Self-Instruct** (Wang et al., 2023), generating 1,500 data points; 7) **LLM2LLM** (Tong et al., 2024a), also generating 1,500 data points; 8) **Rand**, randomly selecting 500 data points per iteration for a total of 1,500; and 9) **LESS** (Xia et al., 2024), selecting 1,500 data points based on gradient similarity.

We adopt the same setting as SEI for self-instruct, except that the sampled examples are selected randomly. Eight samples (five bad cases and three generated data) are selected in each iteration, and GPT-4o generates 20 new samples. This process is repeated to produce a total of 30,000 samples, from which 1,500 training samples are selected using the ICL method. For LLM2LLM and LLMs-as-Instructors, one new sample is generated per bad case using GPT-4o, with 500 samples generated per round over three rounds, resulting in 1,500 samples. We filter out samples with a Rouge-L similarity score above 0.7 during data synthesis

[3] https://github.com/microsoft/unilm/tree/master/mathscale/MWPBench

Models	# Samples	In-Domain		Out-of-Domain				AVG
		GSM8K	MATH	TAL	GaoKao	SAT	College	
Llama-3-8B-Instruct	-	77.56	27.36	37.03	15.55	39.22	15.54	35.38
SEI-FULL	100%	78.01	28.02	38.64	15.94	41.18	**16.25**	36.34
-Rand	5% (1,500)	77.80	**28.54**	37.43	15.16	40.20	15.72	35.81
-LESS	5% (1,500)	77.95	28.18	36.83	14.96	39.22	15.87	35.50
	5% (1,500)	79.76	28.42	**39.91**	**16.73**	**42.15**	15.61	**37.10**
-One-shot ICL	10% (3,000)	**79.98**	27.96	39.37	15.75	40.19	16.22	36.58
	20% (6,000)	79.37	28.18	39.65	15.94	39.22	15.51	36.31
Qwen2.5-Math-7B	-	57.92	50.52	28.07	3.93	39.22	16.96	32.77
SEI-FULL	100%	83.45	60.34	53.57	44.61	67.65	28.22	56.30
-Rand	5% (1,500)	82.52	58.82	53.44	43.58	65.69	27.81	55.31
-LESS	5% (1,500)	83.13	59.76	53.69	45.28	66.67	28.14	56.11
	5% (1,500)	**87.64**	61.28	54.21	**46.06**	**68.62**	28.42	**57.71**
-One-shot ICL	10%(3,000)	85.74	**61.56**	**54.89**	45.76	65.69	28.33	57.16
	20% (6,000)	86.58	60.78	54.76	44.29	63.73	**28.57**	56.45
Mathstral-7B-v0.1	-	80.67	52.58	48.66	47.83	61.76	**25.80**	52.88
SEI-FULL	100%	81.12	53.56	49.13	**49.61**	59.80	25.62	53.14
-Rand	5% (1,500)	79.98	52.50	48.21	47.05	60.78	25.19	52.29
-LESS	5% (1,500)	79.68	52.20	48.60	48.03	60.78	25.23	52.42
	5% (1,500)	82.87	**53.70**	49.47	48.62	62.75	25.72	**53.86**
-One-shot ICL	10% (3,000)	80.52	53.50	48.79	48.23	61.76	24.88	52.95
	20% (6,000)	**83.24**	53.40	**49.53**	46.85	**63.73**	24.77	53.59

Table 3: Model performance under different data selection strategies and samples. The bolded results highlight the best performance achieved using the FULL dataset and the top 5% of samples selected through Rand, LESS, and one-shot ICL methods.

by comparing them against the GSM8K and MATH training and test datasets.

For rand selection, data is proportionally sampled from each error type, with more samples drawn from types with more bad cases. For LESS, following the original setting, we randomly select 10 examples from GSM8K and MATH as the validation set, compute the average gradient of the validation set, and select generated data with the most similar gradients.

5 Experimental Results

5.1 Main Results

Table 2 presents our main results, from which we can draw several conclusions. 1) Our method, SEI-ICL, outperforms others by substantial margins in all math datasets. Specifically, after training, Llama-3-8B-Instruct improves by 1.72% and Mathstral by 0.98%, while Qwen2.5-Math-7B achieves an improvement of 24.94%, highlighting the effectiveness of our error-type-guided data generation approach. 2) Training solely on the original GSM8K and MATH datasets or the identified bad cases results in performance degradation for the Llama3 and Mathstral models. This suggests that existing math training datasets offer limited benefits for already instruction-tuned models. It highlights the necessity of data synthesis. 3) With the same amount of data, our data generation method outperforms other baselines. As shown in Table 2, the average improvement achieved by SEI-ICL on all the models is higher than that of these baselines. Furthermore, combined with the results in Table 3, we observe that even without data selection, randomly selecting the same amount of data (Rand) performs better than self-instruct (random generation), LLMs-as-Instructors and LLM2LLM (based on a single bad case), demonstrating that our error-type-guided data generation is more effective.

5.2 Data Selection

Table 3 presents the results of different data selection methods. By selecting the top 5% of the data using our one-shot learning method, the performance of the trained models on target models surpasses that of SEI-FULL, which uses the full dataset for training. Furthermore, our models con-

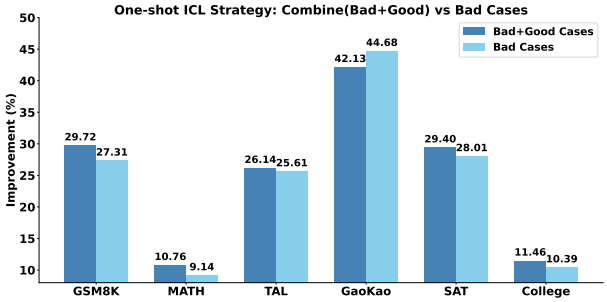

Figure 3: The effects of two one-shot ICL strategies on the improvement of Qwen2.5.

tinue to outperform SEI-FULL as the amount of selected data increases. Under the same data size, the one-shot learning method achieves better results than rand selection and LESS, shows the effectiveness of the one-shot learning approach specifically designed for mathematical problem selection.

We conducted analysis experiments on the data selection validation set $\mathbf{D}_{\text{dev}}$ mentioned in Section 3.3. Specifically, we compared the approach of using only bad cases as $\mathbf{D}_{\text{dev}}$ with the combined approach that includes both good and bad cases. The results of these experiments are shown in Figure 3. It can be observed that the combined approach outperforms the method using only bad cases across most datasets. This demonstrates that, when performing one-shot learning for data selection, it is important to ensure that the generated data addresses bad cases effectively and to maintain the correctness of the original good cases.

5.3 Iterative Improvement Result

	Bad Case (Fix Rate)		Testset (EM Score)	
	GSM8K	**MATH**	**GSM8K**	**MATH**
Iter-0 (ori)	0	0	55.50	32.32
Iter-1	29.98	23.17	79.48	57.21
Iter-2	38.01	39.44	84.70	58.19
Iter-3	**39.13**	**40.57**	**87.79**	**59.18**

Table 4: Bad Case Fix Rate of Qwen2.5-Math on GSM8K and MATH during iterative improvement, along with its performance on the test sets. Bad cases refer to the errors made by Qwen2.5-Math in the training data of GSM8K and MATH.

Table 4 presents the bad case fix rate and test set performance of the Qwen2.5-Math model across different iterations. As shown, with the increase in iterations, the bad case fix rate consistently improves for both datasets, accompanied by a steady improvement in test set performance. This indicates that our method effectively identifies the model's error types in each iteration and generates targeted data for training, thereby enhancing the model's overall performance.

5.4 Iterative vs. From-scratch Training

	GSM8K		MATH	
Model	**Iterative**	**From-scratch**	**Iterative**	**From-scratch**
Llama3	78.09	**79.76**	27.62	**28.42**
Qwen2.5	**87.79**	87.64	59.18	**61.28**
Mathstral	81.96	**82.87**	48.02	**53.70**

Table 5: Comparison of model performance on GSM8K and MATH tasks under different training methods (Iterative and From-scratch).

Table 5 highlights the differences between iterative training and from-scratch training within our framework. In iterative training, each new iteration continues training the target model obtained in the previous round. In contrast, from-scratch training involves directly training the initial target model once the data is obtained after three rounds of data generation. The results show that from-scratch training outperforms iterative training. A possible explanation for this is that in each round of iterative training, we only select the top 5% of the data for training. With such a small amount of data, iterative fine-tuning may lead to overfitting over multiple rounds. On the other hand, training from scratch aggregated datasets helps mitigate this issue, resulting in better overall performance.

5.5 Different Synthetic Size

We conducted an analysis between the amount of unfiltered synthetic data and performance, with the results presented in Figure 4. It can be observed that for all target models, the size of the generalization data is not proportional to performance. For Llama3, performance initially improves but eventually starts to decline. Specifically, the best performance on GSM8K is achieved with 15,000 training samples, while the optimal result on MATH is reached with 25,000 samples. In contrast, the results for Qwen2.5 and Mathstral are relatively inconsistent. These findings further highlight the importance of data selection. For models like Llama3 and Mathstral, which have already undergone extensive instruction tuning, the quantity of data may not be the key to improving performance. Instead, the focus should shift to constructing small but high-quality datasets.

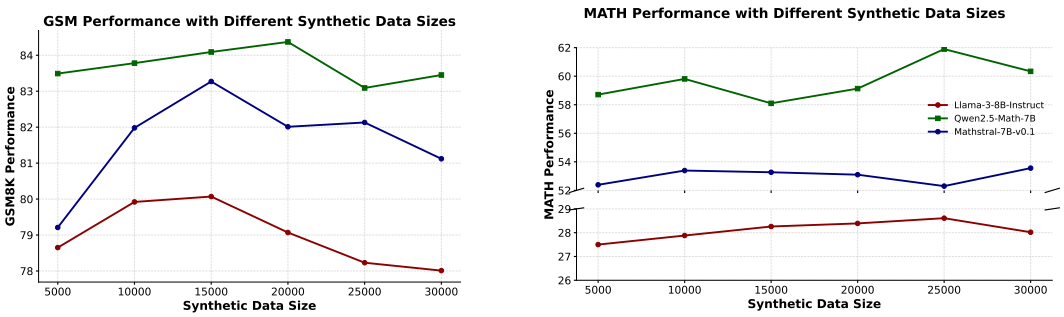

Figure 4: Comparison of GSM8K and MATH performance under different synthetic data sizes.

6 Conclusion

We propose Self-Error-Instruct, a novel framework to improve LLMs mathematical reasoning by generalizing training data based on error types rather than individual bad cases. Our method enhances data diversity and mitigates overfitting by analyzing errors, clustering them into categories, and synthesizing targeted data using a self-instruct approach. Experiments on LLaMA3-8B-Instruct, Qwen2.5-Math-7B, and Mathstral demonstrate notable performance improvements with our method, achieving average gains of 1.72%, 24.94%, and 0.98%, respectively, across both in-domain and out-of-domain evaluations.

Limitations

Our framework has three main limitations: the high cost of using GPT-4o as the instructor model, the focus on GSM8K and MATH datasets for bad case extraction, which may limit the diversity of errors, and the increased time consumption caused by one-shot learning.

Our approach is the reliance on GPT-4o as the instructor model for error analysis and data synthesis. While GPT-4o is highly effective in identifying error keyphrases and generating targeted training data, its use incurs significant computational and financial costs, which may limit the scalability and accessibility of the framework.

The second limitation lies in the scope of our bad case extraction and iterative refinement process, which is currently confined to the GSM8K and MATH datasets. As a result, the error types identified and addressed may be limited to those specific to these datasets, potentially restricting the generalizability of the framework to other mathematical reasoning tasks or datasets. In the future, a more dynamic approach could be adopted, where bad cases are extracted from the initial datasets and continuously identified within the synthesized data during the iterative process. This would allow the framework to discover new and diverse error types as the training data evolves, further broadening the issues addressed and enhancing the model's mathematical reasoning capabilities. This expansion would help ensure the framework adapts to various problems, improving its robustness and applicability to real-world scenarios.

The third limitation lies in the one-shot data selection process. Although this approach is a one-time operation and produces results superior to LESS and random selection, the one-shot learning phase requires significant computational resources. This is because each of the 30,000 generated samples needs to be validated against an ICL-formatted validation set containing 100 samples.

Ethics Considerations

This study strictly uses OpenAI's GPT-4o model for research purposes, in compliance with OpenAI's Business Terms, Section 2-(e). Our work analyzes reasoning errors to improve AI models and does not involve developing or commercializing competing products. We ensure no derived models are distributed or made available to third parties, maintaining full adherence to ethical and legal standards.

Acknowledgements

This work is supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. PolyU/25200821), the Innovation and Technology Fund (Project No. PRP/047/22FX), PolyU Internal Fund from RC-DSAI (Project No. 1-CE1E), and a gift fund from Huawei (N-ZGM3).

References

Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. 2024. Learning from mistakes makes llm better reasoner. *Preprint*, arXiv:2310.20689.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Lingjie Chen, Ruizhong Qiu, Siyu Yuan, Zhining Liu, Tianxin Wei, Hyunsik Yoo, Zhichen Zeng, Deqing Yang, and Hanghang Tong. 2024. Wapiti: A watermark for finetuned open-source llms. *arXiv preprint arXiv:2410.06467*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Fe-

ichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Anumanchipalli, Michael Mahoney, Kurt Keutzer, and Amir Gholami. 2024. LLM2LLM: Boosting LLMs with novel iterative data enhancement. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6498–6526, Bangkok, Thailand. Association for Computational Linguistics.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024a. From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7602–7635, Mexico City, Mexico. Association for Computational Linguistics.

Yunshui Li, Binyuan Hui, Xiaobo Xia, Jiaxi Yang, Min Yang, Lei Zhang, Shuzheng Si, Ling-Hao Chen, Junhao Liu, Tongliang Liu, Fei Huang, and Yongbin Li. 2024b. One-shot learning as instruction data prospector for large language models. In *Proceedings of the 62nd Annual Meeting of the Association*

for Computational Linguistics (Volume 1: Long Papers), pages 4586–4601, Bangkok, Thailand. Association for Computational Linguistics.

Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. On llms-driven synthetic data generation, curation, and evaluation: A survey. *Preprint*, arXiv:2406.15126.

OpenAI. 2024a. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

OpenAI. 2024b. Gpt-4o.

OpenAI. 2024. O1 Model. https://openai.com/o1/. Accessed: 2024-12-11.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

TAL. 2023. Tal-scq5k. https://github.com/math-eval/TAL-SCQ5K. GitHub repository.

Ruixiang Tang, Xiaotian Han, Xiaoqian Jiang, and Xia Hu. 2023. Does synthetic data generation of llms help clinical text mining? *Preprint*, arXiv:2303.04360.

Zhengyang Tang, Xingxing Zhang, Benyou Wang, and Furu Wei. 2024. Mathscale: Scaling instruction tuning for mathematical reasoning. *Preprint*, arXiv:2403.02884.

Gemini Team. 2024. Gemini: A family of highly capable multimodal models. *Preprint*, arXiv:2312.11805.

Terry Tong, Qin Liu, Jiashu Xu, and Muhao Chen. 2024a. Securing multi-turn conversational language models from distributed backdoor attacks. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12833–12846, Miami, Florida, USA. Association for Computational Linguistics.

Yongqi Tong, Dawei Li, Sizhe Wang, Yujia Wang, Fei Teng, and Jingbo Shang. 2024b. Can LLMs learn from previous mistakes? investigating LLMs' errors to boost for reasoning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3065–3080, Bangkok, Thailand. Association for Computational Linguistics.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Zhepei Wei, Wei-Lin Chen, and Yu Meng. 2025. InstructRAG: Instructing retrieval-augmented generation via self-synthesized rationales. In *The Thirteenth International Conference on Learning Representations*.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. LESS: Selecting influential data for targeted instruction tuning. In *International Conference on Machine Learning (ICML)*.

Ran Xu, Hejie Cui, Yue Yu, Xuan Kan, Wenqi Shi, Yuchen Zhuang, May Dongmei Wang, Wei Jin, Joyce Ho, and Carl Yang. 2024. Knowledge-infused prompting: Assessing and advancing clinical text data generation with large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 15496–15523, Bangkok, Thailand. Association for Computational Linguistics.

Boyang Xue, Fei Mi, Qi Zhu, Hongru Wang, Rui Wang, Sheng Wang, Erxin Yu, Xuming Hu, and Kam-Fai Wong. 2024a. Ualign: Leveraging uncertainty estimations for factuality alignment on large language models. *Preprint*, arXiv:2412.11803.

Boyang Xue, Hongru Wang, Rui Wang, Sheng Wang, Zezhong Wang, Yiming Du, Bin Liang, and Kam-Fai Wong. 2024b. A comprehensive study of multilingual confidence estimation on large language models. *Preprint*, arXiv:2402.13606.

Boyang Xue, Weichao Wang, Hongru Wang, Fei Mi, Rui Wang, Yasheng Wang, Lifeng Shang, Xin Jiang, Qun Liu, and Kam-Fai Wong. 2023. Improving factual consistency for knowledge-grounded dialogue systems via knowledge enhancement and alignment. *Preprint*, arXiv:2310.08372.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *Preprint*, arXiv:2409.12122.

Jiahao Ying, Mingbao Lin, Yixin Cao, Wei Tang, Bo Wang, Qianru Sun, Xuanjing Huang, and Shuicheng Yan. 2024. LLMs-as-instructors: Learning from errors toward automating model improvement. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11185–11208, Miami, Florida, USA. Association for Computational Linguistics.

Erxin Yu, Jing Li, Ming Liao, Siqi Wang, Gao Zuchen, Fei Mi, and Lanqing Hong. 2024a. CoSafe: Evaluating large language model safety in multi-turn dialogue coreference. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17494–17508, Miami, Florida, USA. Association for Computational Linguistics.

Erxin Yu, Jing Li, and Chunpu Xu. 2024b. PopALM: Popularity-aligned language models for social media trendy response prediction. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 12867–12878, Torino, Italia. ELRA and ICCL.

Erxin Yu, Jing Li, and Chunpu Xu. 2024c. RePALM: Popular quote tweet generation via auto-response augmentation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 9566–9579, Bangkok, Thailand. Association for Computational Linguistics.

Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. 2024. Evaluating the performance of large language models on gaokao benchmark. *Preprint*, arXiv:2305.12474.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2024. AGIEval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2299–2314, Mexico City, Mexico. Association for Computational Linguistics.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment. *Preprint*, arXiv:2305.11206.

A Overview of Prompts Used

A.1 Prompt for Training and Inference

For all the models, we use the built-in chat templates for training and inference. Figure 7 illustrates the one-shot learning prompt for the Qwen2.5 model, where the model generates a response by being presented with an example of a synthetic question paired with its solution.

A.2 Prompt for Error Keyphrase Generation

Figure 5 illustrates the prompt used to generate error keyphrases for identifying and summarizing mistakes in mathematical reasoning. The input to the prompt includes a math question, the correct reasoning path leading to the answer, and the model's incorrect reasoning path. The prompt instructs the model to analyze where the error occurred in its reasoning process, identify the cause, and summarize it as a concise yet descriptive keyphrase. The output is a single keyphrase in list format, effectively capturing the primary reason for the model's mistake, which can then be used for further error analysis and targeted data synthesis.

A.3 Prompt for Error Clustering Generation

Figure 6 presents a prompt designed to guide the analysis and categorization of error keyphrases generated from a model's reasoning mistakes. The input to this prompt is a list of error keyphrases, and the task involves clustering these keyphrases based on common themes, causes, or areas of occurrence. For each cluster, the model is instructed to list the included keyphrases, explain their grouping, and assign a concise, descriptive name to the cluster. This process helps identify patterns in the model's errors, offering meaningful insights into the types of mistakes made and enabling targeted improvements in the model's reasoning capabilities.

A.4 Prompt for Error Type-Specific Data Synthesis

The prompt in Figure 8 and 9 guides the creation of 20 challenging math problems targeting specific error types in the GSM8K and MATH datasets. By analyzing the examples provided, the instruct model identifies patterns or issues causing errors and generates diverse, difficult problems aligned with these error types. The output follows a strict JSON format with detailed solutions and final numerical answers.

B Related Work on Data Synthesis

The generation of synthetic data driven by large language models has become an essential method for addressing the issues of data quantity and quality in the field of deep learning (Long et al., 2024). LLMs, with their powerful language understanding and generation capabilities, can produce synthetic data that closely resembles the characteristics and patterns of real-world data (Wang et al., 2023). This synthetic data can not only serve as a substitute or supplement for real data but can also be generated according to specific instructions and conditions to meet the needs of different applications (Yu et al., 2024c). The use of LLM-driven synthetic data generation is widespread across various fields, including general alignment (Chen et al., 2024; Yu et al., 2024a; Xue et al., 2024a), mathematical reasoning (Lee et al., 2024; Ying et al., 2024), medical (Tang et al., 2023; Xu et al., 2024),

Error Keyphrase Generation Prompt:

Based on the given mathematical problem, identify the step where the model made an error in its reasoning process. Analyze the reason for this error and summarize it using a keyphrase. The input consists of a math question, the correct answer, and the model's incorrect answer. Please output the result in the following format:

[”Error keyphrase"]

Ensure that your analysis focuses on the mistake in the model's problem-solving process. The keyphrases should be concise yet descriptive, effectively summarizing the primary reason for the model's mistake. Strictly adhere to the list format output without any additional information.

Math Question: {Question $\boldsymbol{q_i}$}
Answer: {Correct Reasoning Path $\boldsymbol{r_i}$ }
Model Output: {Incorrect Model Reasoning Path $\boldsymbol{\hat{r}_i}$}

Figure 5: Prompt for Generating Error Keyphrases.

Error Keyphrases Clustering Prompt:

You are an expert in error analysis and categorization. You will be given a list of error keyphrases. Your task is to:

1. Analyze the given error keyphrases and identify common themes or patterns.
2. Group similar keyphrases together based on their likely causes, effects, or areas of occurrence.
3. For each cluster:
    a. List the keyphrases in the cluster.
    b. Explain why these keyphrases are grouped together.
    c. Assign a concise but descriptive name to the cluster that captures its essence.
    4. Clusters should cover all the keyphrases.
5. Present your results in a clear, structured format.

Strictly output in plain text according to the following format, do not output in other formats or with extra symbols:
[
{{"Cluster name":, "Keyphrases":[], "explanation":,}},
{{"Cluster name":, "Keyphrases":[], "explanation":,}} ...
]

Your clustering should aim to provide meaningful insights that can help in understanding and addressing the errors more effectively.
Here is the list of error keyphrases: {Error Keyphrases Set **E-set**}

Figure 6: Prompt for Clustering Error Keyphrases

Ono–shot Learning Prompt:

Please reason step by step, and put your final answer within \boxed{}.

Here is an example:
Instruction: {Synthetic Question from $\boldsymbol{D_{SEI}}$}
Response: {Synthetic Solution from $\boldsymbol{D_{SEI}}$}

Instruction: {Question from $\boldsymbol{D_{dev}}$}
Response:

Figure 7: One-Shot Learning Prompt for Selecting Synthetic Data

Error Type-Specific Data Synthesis for GSM8K:

Based on the given examples and error type, create 20 difficult math problems that are likely to cause errors in the model.

Requirement:
1. Identify the commonality in the given examples and consider what issues in these examples might cause the model to make mistakes.
2. Make the new problems more challenging and diverse.
3. Format the output strictly as a string in this structure:
[{{"question":,"solution":}},
{{"question":,"solution":,}},...].
Ensure no additional output beyond the specified structure. Output in JSON format.
4. The reasoning process for each step should be provided in the solution.
5. Ensure the final answer is a number and place it on a new line, denoted by \n#### num.
6. Don't make any mathematical mistakes of your own!

Provided Questions:
{Sampled Error Question $\boldsymbol{q_1}$}
{Sampled Error Question $\boldsymbol{q_2}$}
{Sampled Error Question $\boldsymbol{q_3}$}
{Sampled Error Question $\boldsymbol{q_4}$}
{Sampled Error Question $\boldsymbol{q_5}$}
{Sampled Error Question $\boldsymbol{q_6}$}
{Sampled Error Question $\boldsymbol{q_7}$}
{Sampled Error Question $\boldsymbol{q_8}$}
Error Type:
{Error type}
Generated Questions:

Figure 8: Prompt for GSM8K Error Type-Specific Data Synthetic.

Error Type-Specific Data Synthesis for Math:

Based on the given examples and error type, create 20 difficult math problems that are likely to cause errors in the model.

Requirement:
1. Identify the commonality in the given examples and consider what issues in these examples might cause the model to make mistakes.
2. Make the new problems more challenging and diverse.
3. Format the output strictly as a string in this structure:
[{{"question":,"solution":}},
{{"question":,"solution":,}},...].
Ensure no additional output beyond the specified structure. Output in JSON format.
4. The reasoning process for each step should be provided in the answer.
5. The final answer should be marked with \\boxed{{}}
When generating math problems in JSON format:
1) Use \\\\(and \\\\) for inline math
2) Avoid complex LaTeX commands
3) Use simple alternatives for arrows and dots
4) Keep solutions concise and avoid unnecessary formatting
5) Escape special characters properly
6) Test the JSON validity before finalizing
6.Don't make any mathematical mistakes of your own!

Provided Questions:
{Sampled Error Question $\boldsymbol{q_1}$}
{Sampled Error Question $\boldsymbol{q_2}$}
{Sampled Error Question $\boldsymbol{q_3}$}
{Sampled Error Question $\boldsymbol{q_4}$}
{Sampled Error Question $\boldsymbol{q_5}$}
{Sampled Error Question $\boldsymbol{q_6}$}
{Sampled Error Question $\boldsymbol{q_7}$}
{Sampled Error Question $\boldsymbol{q_8}$}
Error Type:
{Error type}
Generated Questions:

Figure 9: Prompt for MATH Error Type-Specific Data Synthetic.

social media (Wei et al., 2025; Yu et al., 2024b), and hallucination (Xue et al., 2024b, 2023).

C Manual Category Review

We applied two manual adjustments after clustering: **merging categories** and **excluding categories**.

During the clustering process, some duplicate or similar categories may be generated, such as "Timezone and Duration Calculation Errors" and "Time and Duration Calculation Errors," or "Calculation Errors" and "General Calculation Errors." These categories essentially represent the same or closely related error types, so we merge them into a unified category to avoid redundancy.

We identify bad cases by comparing the model's extracted answers with the correct ones. However, this method may lead to a small number of correct answers being mistakenly identified as errors, which is a common issue in math evaluations. Fortunately, GPT-4o is usually able to determine that these responses are actually correct. Consequently, a special category like "No Error" or "Correct Process" may appear after clustering, and we manually exclude this category because it does not represent actual error types. Through these manual reviews, we can more accurately organize and analyze error categories, ensuring the reliability and consistency of the results.